How to: unpacking di PC Guard 5.0

Data	by "epokh"	ï¿½
08/03/2005	***UIC's Home Page***	**Published by Quequero**
Per il fatto che l'inchiostro è soprattutto acqua, la calligrafia cinese, nel suo controllare lo scorrere dell'acqua	*Ho colpito il tuo tsubo chiamato Seichi Nobok, da questo momemto in poi proverai una voglia irrefrenabile di ucciderti di seghe.....* *Scusa Que ma non c'è lo fatta a resistere prima o poi dovevo dirla questa frase!* ***Inclusi gli orrori di ortografia? :P Grazie epo, splendido tute! Poi mi fai giocare con quel coso quel vola? :)))***	con il pennello morbido in quanto distinto dalla penna dura, richiede che si segua la corrente. Alan W.Watts
....	**E-mail:** epokh@libero.it	....
Difficoltï¿½	( )**NewBies** ()**Intermedio** (X)**Avanzato** ( )**Master**	ï¿½

ï¿½

Introduzione

In questo tutorial verrà esposto un metodo **generale** per unpackare programmi packati (scusate il gioco di parole) con il nuovo PC Guard 5.0 .
PC Guard v 5.0 è un programma scritto da Blagoje Ceklic per proteggere programmi per Windows 95/98/ME/2000/NT/XP/2003 e per .NET dal reverse engineering e da distribuzioni illegali (pensate un pò).
Poi unpackeremo con il metodo descritto, due programmi: uno scritto da me da cui ho presp alcuni screenshoot e poi lo stesso applicativo PC Guard 5.0 che serve appunto per proteggere i programmi.
Un bell'esempio di tutorial altamente ricorsivo!

Tools usati

OllyDebugger con i seguenti plugin:
OllyDump
IsDebugPresent
L'ottimo ImportReconstructor 1.6
PEiD

URL o FTP del programma

PC Guard 5.0

Notizie sul programma

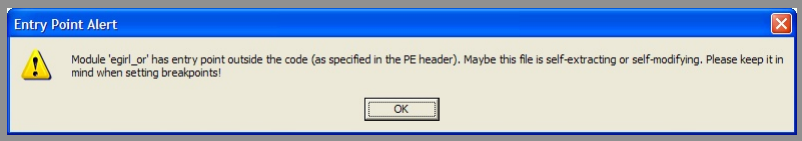
Unpackeremo lo stesso programma PC Guard 5.0 protetto in modo ricorsivo da se stesso.

Essay

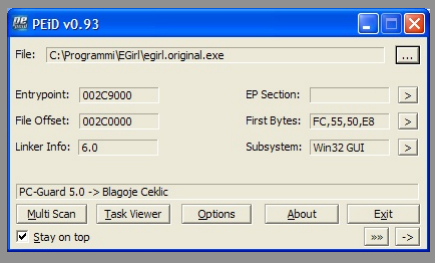
Le modalità di protezione offerte da PC Guard 5.0 sono 4: remote, code, plain,network.
Modalità Remote
wrapping + encryption + anti-debugging + locking + activation keys
Il programma è cifrato e wrappato. Un codice di attivazione valido è richiesto per ogni computer che si vuole sbloccare.
Modalità Code
wrapping + encryption + anti-debugging + locking
Il programma è cifrato e wrappato. Il programma si sblocca dal codice ottenuto da un computer remoto.
Il programma è vincolato al target computer
Modalità Plain
wrapping + encryption + anti-debugging
Il programma è cifrato e wrappato. Il programma non è vincolato al target computer e non è richiesta nessuna chiave di attivazione.
Modalità Network
wrapping + encryption + anti-debugging + network licensing
Il programma è cifrato e wrappato. Solo un numero limitato di workstation su una rete possono accedere al programma protetto allo stesso tempo.

Le prove che ho fatto io sono state con la modalità Plain, anche se sembra funzionare anche con le altre modalità ma non ne sono certo al 100%.

La prima verifica è se il nostro programma da reversare è packato con PC Guard 5.0, questa in genere può essere fatta sia con PEId oppure anche con OllyDump.
OllyDump ci avverte che il programma è packato con il messaggio seguente:



Mentre il PEId identifica che il Programma è packato con PC Guard in questo modo:

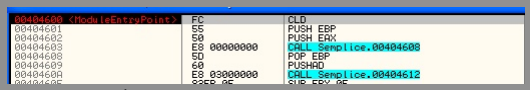


Una volta confermato che il programma è stato packato con PC Guard procediamo con il suo unpacking.
Il metodo per l'unpacking di PC Guard è diviso nei seguenti passi:

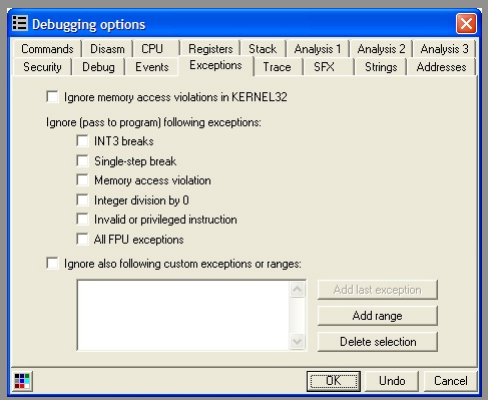
a)Trovare l'OEP
b)Fare il dump del processo debuggato
c)Ricostruire la IAT

Trovare l'OEP

Apriamo il programma che ci interessa unpackare con OllyDump e mettiamo subito un breackpoint sell'entry point del programma così:



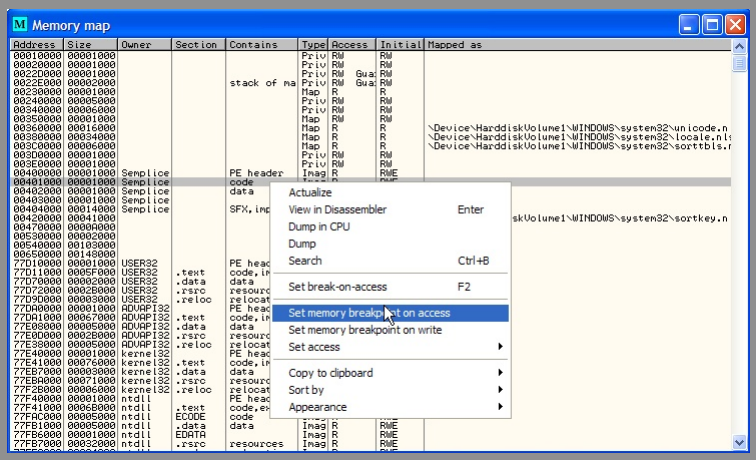
Per trovare l'OEP è necessario tracciare tutte le eccezioni che il programma genererà durante l'esecuzione,quindi nelle opzioni del debugger deselezioniamo tutto, in modo che OllyDB non ignori nessuna eccezione :



Ora PRIMA di iniziare il debug del processo nascondiamo il debugger con il plugin IsDebugPresent->Hide e iniziamo il debugging con F9.
Durante l'esecuzione basterà premere Shift+F9 per bypassare l'eccezione, ma attenzione perchè l'OEP si trova prima dell'ultima eccezione, e quindi bisogna stare attenti a non superarla.
Lo schema generale da seguire per trovare l'OEP è il seguente:

- All'inizio: Access Violation in [XXXXXXXX]
- Un altra Access Violation in [XXXXXXXX]
- Un Single Step
- Un altro Single Step
- Un INT3 su ntdll.DbgBreakPoint
- Un secondo INT3 su ntdll.DbgBreakPoint
- Un terzo INT3 su ntdll.DbgBreakPoint
- Un Single Step
  L'OEP si trova tra questa eccezione e la seguente
- Un ultimo Single Step

Nel momento in cui arriviamo alla **penultima eccezione Single Step** basta mettere un **memory breakpoint on access** sulla sezione codice del programma in debugging come in figura:



E poi continuiamo a bypassare le eccezioni fino a quando non arriviamo al nostro memory breakpoint il cui indirizzo sarà l'OEP tanto atteso!
Ora di seguito riporto alcuni log di OllyDbg, colorati in modo da raggruppare lo schema dell eccezioni descritto sopra, che mostrano (quasi) tutti i casi possibili che vi possono capitare.

Il comportamento più semplice è il seguente:

```
IsDebugPresent hidden <<-- nascondo il debugger
004048DE Access violation when writing to [657EAB6E] <<-- il blocco delle eccezioni
00404AFF Access violation when writing to [B4E6507E]
00404C73 Single step event at Semplice.00404C73
77F65A58 INT3 command at ntdll.DbgBreakPoint
77F65A58 INT3 command at ntdll.DbgBreakPoint
77F65A58 INT3 command at ntdll.DbgBreakPoint
004117C6 Single step event at Semplice.004117C6
004160C4 Single step event at Semplice.004160C4
00401220 Memory breakpoint when executing [00401220] <<-- ecco il nostro OEP
```

Programma che carica DLL fra l'ultimo ed il penultimo single step:

```
0060E000   Program entry point
             IsDebugPresent hidden
0060E343   Access violation when writing to [657EAB6E]
0060E564   Access violation when writing to [B4E6507E]
0060E6D8   Single step event at <programma>.0060E6D8
0060EA0D   Single step event at <programma>.0060EA0D
77F65A58   INT3 command at ntdll.DbgBreakPoint
77F65A58   INT3 command at ntdll.DbgBreakPoint
77F65A58   INT3 command at ntdll.DbgBreakPoint
0061D999   Single step event at <programma>.006
003A0000   Module C:\cartella\<programma>\<programma>.DLL
76610000   Module C:\WINDOWS\System32\SETUPAPI.dll
77BE0000   Module C:\WINDOWS\system32\msvcrt.dll
76340000   Module C:\WINDOWS\system32\comdlg32.dll
70A70000   Module C:\WINDOWS\system32\SHLWAPI.dll
78090000   Module C:\WINDOWS\WinSxS\...\COMCTL32.dll
77390000   Module C:\WINDOWS\system32\SHELL32.dll
0062198F   Single step event at <programma>.0062198F
72F50000   Module C:\WINDOWS\System32\WINSPOOL.DRV
004475FC   Memory breakpoint when executing [004475FC]  <<--- OEP!
```

In giallo possiamo notare i moduli importati dal programma fra l'ultima e la penultima eccezione Single Step.

Programma che carica una DLL packata con PC Guard 5.0:
Nei log seguenti possiamo notare come venga caricata una DLL packata con PC Guard:

```
00570000   Program entry point
00570349   Access violation when writing to [657EAB6E] <-- il blocco di eccezioni dell'eseguibile
0057056A   Access violation when writing to [B4E6507E]
005706DE   Single step event at <programma>.005706DE
00570A13   Single step event at <programma>.00570A13
77F65A58   INT3 command at ntdll.DbgBreakPoint
77F65A58   INT3 command at ntdll.DbgBreakPoint
00390000   Module C:\programma\<unaDllpackata>.dll <-- la dll viene caricata qui
             Code section extended to include self-extractor
00398343   Access violation when writing to [657EAB6E] <-- il blocco di eccezioni della DLL
00398564   Access violation when writing to [B4E6507E]
003986D8   Single step event at <unaDllpackata>.003986D8
00398A0D   Single step event at <unaDllpackata>.00398A0D
77F65A58   INT3 command at ntdll.DbgBreakPoint
77F65A58   INT3 command at ntdll.DbgBreakPoint
77F65A58   INT3 command at ntdll.DbgBreakPoint
003A79FD   Single step event at <unaDllpackata>.003A79FD
77390000   Module C:\WINDOWS\system32\SHELL32.dll
77BE0000   Module C:\WINDOWS\system32\msvcrt.dll
70A70000   Module C:\WINDOWS\system32\SHLWAPI.dll
78090000   Module C:\WINDOWS\WinSxS\...\comctl32.dll
003AB9F3   Single step event at <unaDllpackata>.003AB9F3
77300000   Module C:\WINDOWS\system32\comctl32.dll
00AD0000   Module C:\Program Files\Windows Privacy Tools\WinPT\PTD.dll
5B090000   Module C:\WINDOWS\system32\uxtheme.dll
00C00000   Module C:\Program Files\Windows Privacy Tools\GPGOE\GPGOE.dll
00D30000   Module C:\blackbox\plugins\BBLeanSkin\bbLeanSkinEng.dll
00E20000   Module C:\Program Files\Apoint2K\EzAuto.dll
003AB9F3   Single step event at <unaDllpackata>.003AB9F3
00390000   Unload C:\programma\<unaDllpackata>.dll <-- la dll viene scaricata qui
77F65A58   INT3 command at ntdll.DbgBreakPoint
0057FABB   Single step event at <programma>.0057FABB
00F40000   Module C:\WINDOWS\System32\PEGRP32C.dll
76340000   Module C:\WINDOWS\system32\comdlg32.dll
72F50000   Module C:\WINDOWS\System32\WINSPOOL.DRV
76310000   Module C:\WINDOWS\System32\msimg32.dll
74CA0000   Module C:\WINDOWS\System32\oledlg.dll
7CCC0000   Module C:\WINDOWS\system32\ole32.dll
5F140000   Module C:\WINDOWS\System32\OLEPRO32.DLL
00583B42   Single step event at <programma>.00583B42 <-- parte finale del blocco dell'eseguibile
770E0000   Module C:\WINDOWS\system32\OLEAUT32.dll
0049AD55   Memory breakpoint when executing [0049AD55] <<--- OEP !
```

La DLL packata unaDllpackata.dll genera lo stesso schema di eccezioni dei normali eseguibili.

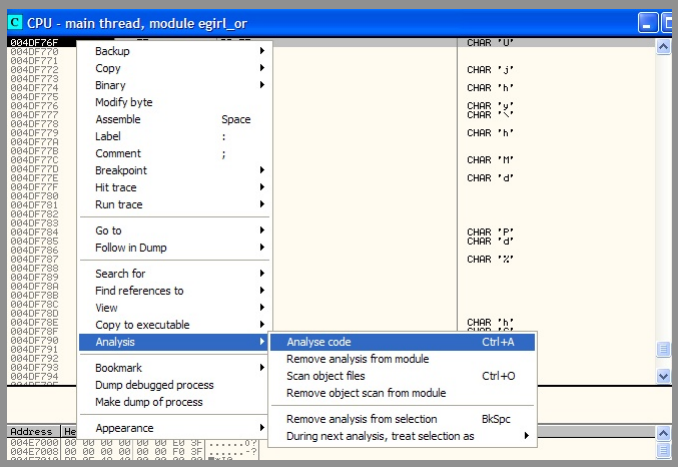
Programma packato con PC Guard 2 volte!
Il log seguente mostra come il programma in esame sia stato packato 2 volte con PC Guard, visto che lo schema delle eccezioni si ripete 2 volte:
la prima volta in giallo e la seconda volta in verde:

```
0048A000   Program entry point
             IsDebugPresent hidden
0048A349   Access violation when writing to [657EAB6E] <-- 1' blocco di eccezioni
0048A56A   Access violation when writing to [B4E6507E]
0048A6DE   Single step event at <programma>.0048A6DE
0048AA23   Single step event at <programma>.0048AA23
77F65A58   INT3 command at ntdll.DbgBreakPoint
77F65A58   INT3 command at ntdll.DbgBreakPoint
77F65A58   INT3 command at ntdll.DbgBreakPoint
0049A4F2   Single step event at <programma>.0049A4F2
0049E6BE   Single step event at <programma>.0049E6BE
00474349   Access violation when writing to [657EAB6E] <-- 2' blocco di eccezioni
0047456A   Access violation when writing to [B4E6507E]
004746DE   Single step event at <programma>.004746DE
00474A23   Single step event at <programma>.00474A23
77F65A58   INT3 command at ntdll.DbgBreakPoint
77F65A58   INT3 command at ntdll.DbgBreakPoint
77F65A58   INT3 command at ntdll.DbgBreakPoint
77314AF2   Single step event at <programma>.00494AF2
77300000   Module C:\WINDOWS\system32\COMCTL32.dll
76C40000   Module C:\WINDOWS\system32\IMAGEHLP.dll
77BE0000   Module C:\WINDOWS\system32\msvcrt.dll
76340000   Module C:\WINDOWS\system32\comdlg32.dll
70A70000   Module C:\WINDOWS\system32\SHLWAPI.dll
77390000   Module C:\WINDOWS\system32\SHELL32.dll
78090000   Module C:\WINDOWS\WinSxS\...\comctl32.dll
004886BE   Single step event at <programma>.004886BE
77BD0000   Module C:\WINDOWS\system32\VERSION.dll
00428238   Memory breakpoint when executing [00428238] <<--- OEP !
```

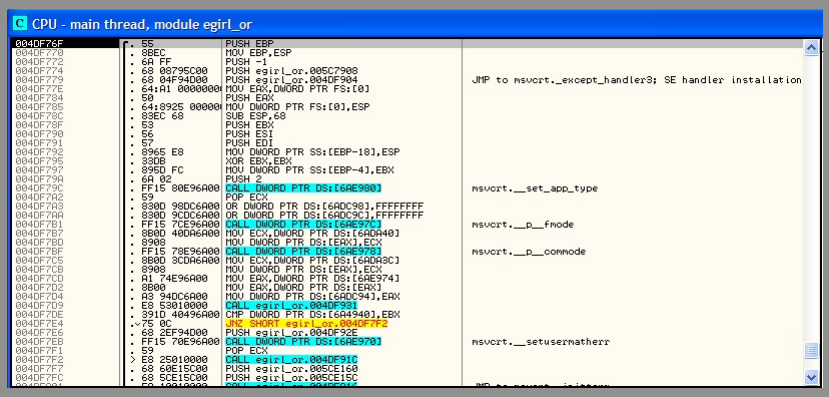
Ora non resta che fare il dump del processo in debug.

Dump del processo

Il dump del processo va fatto a partire dall'OEP precedentemente trovato nei Log di OllyDB. Per controllare se l'OEP è corretto possiamo anche analizzare il codice a partite dall'OEP mediante la funzione Analyse code:

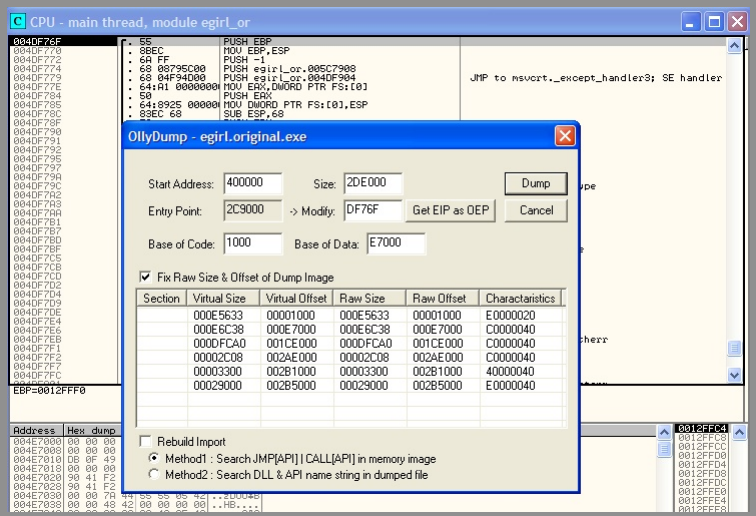


Che nell'esempio in questione produce:



Come possiamo vedere siamo sulla buona strada visto che le prime istruzioni sono quelle per l'attivazione di un nuovo record di attivazione.

Poi usiamo come al solito OllyDump,e facciamo il dump del processo senza ricostruire la IAT, operazione che faremo in seguito con ImportRec . Ecco uno screenshoot significativo:

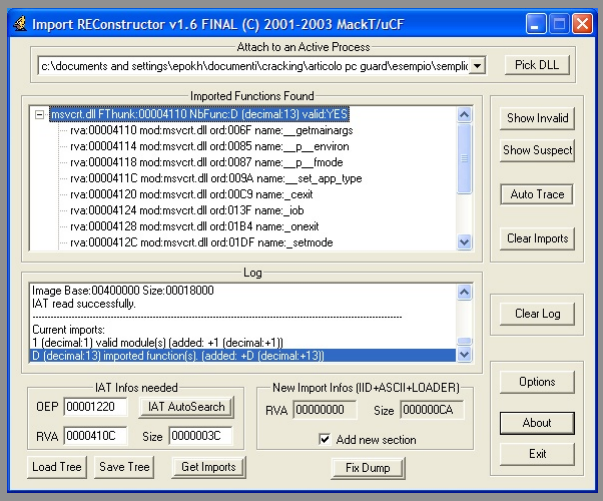


Poi ricordiamoci l'indirizzo relativo dell'OEP e non quello assoluto: nel caso in questione *l'indirizzo relativo dell'OEP è DF76F* a partire da 400000, mentre quello assoluto è 4DF76F.

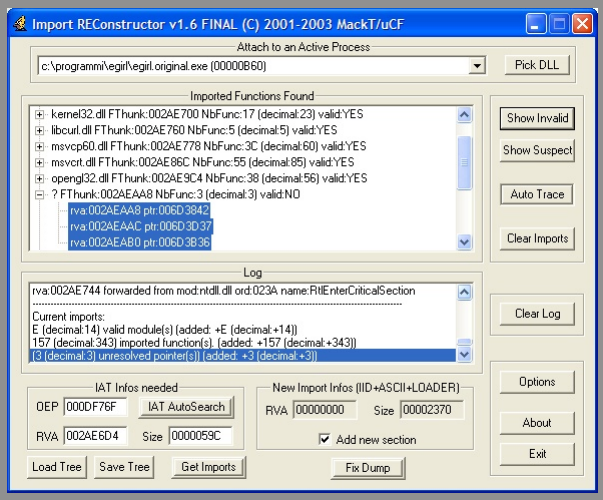
Ricostruzione della IAT

Ora bisogna "solo" ricostruire la IAT (=Import Address Table) del processo di cui abbiamo fatto il dump. Lanciamo ImportRec ed eseguiamo **il programma originale.** Poi attacchiamo ImportRec al programma originale in esecuzione e inseriamo il valore dell'OEP precedentemente trovate e poi troviamo la IAT mediante **IATAutoSearch** e visualizziamo le funzioni importate importate con il comando **GetImports.**

Ora possono verificarsi 2 casi :
a) il caso fortunato: tutti i puntatori sono risolti allora basta fixare il dump del programma originale:



b) il caso sfortunato: ci sono alcuni puntatori irrisolti in genere ho notato che basta fixarli con le seguenti funzioni:
kernel32.ExitProcess,kernel32.ExitThread, msvcrt._exit
Come nel caso qui sotto,facendo ShowInvalid si notano tre puntatori non risolti:

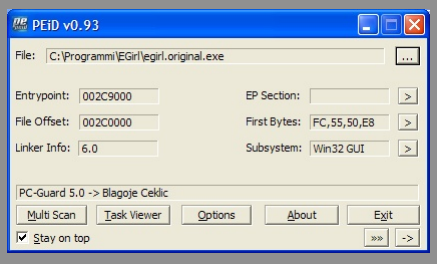


Che fixiamo con le opportune ExitProcess di Kernel32.Se questo non dovesse funzionare, siamo abbastanza nella m***a perchè dovremmo andare a spulciare fra le altre librerie importate e vedere quale funzione manca basandosi sul proprio intuito o culo.

NOTA:
Vi ricordo, visto che accade spesso, che quando fixiamo il dump dobbiamo selezionare il dump del processo originale e non l'exe packato!
A questo punto non ci resta che crackare il nostro bel programma unpackato come siamo soliti fare!

Unpacking di PC Guard 5.0

Vediamo di applicare le tecniche sopra esposte per unpackare un programma protetto con PC Guard 5.0. E quale programma ho scelto secondo voi?
Lo stesso PC Guard 5.0 che indovinate con cosa è packato? Eh si proprio con se stesso? (Mi chiedo come abbiano fatto: è uguale al problema dell'uovo e della gallina).
Infatti il nostro fido PEid ci segnala giustamente che:



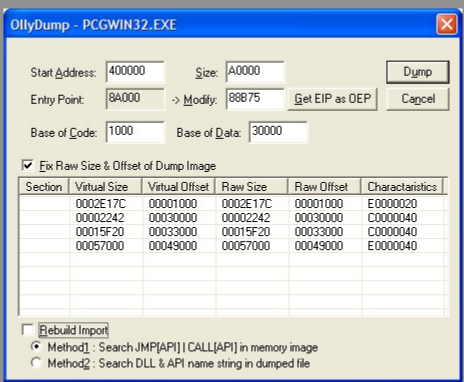
Allora carichiamo PCGWIN32.EXE con Ollydbg e troviamo l'OEP con i metodi descritti in precedenza. In particolare bypassiamo tutte le eccezioni (Shift+F9) in modo da eseguirlo fino al suo caricamento e poi chiudiamo PC Guard. I Log di Ollydbg sono abbastanza eloquenti:

```
IsDebugPresent hidden
0048A349 Access violation when writing to [657EAB6E] <<-- 1' blocco di eccezioni
0048A56A Access violation when writing to [B4E6507E]
0048A6DE Single step event at PCGWIN32.0048A6DE
0048AA23 Single step event at PCGWIN32.0048AA23
77F65A58 INT3 command at ntdll.DbgBreakPoint
77F65A58 INT3 command at ntdll.DbgBreakPoint
77F65A58 INT3 command at ntdll.DbgBreakPoint
0049A4F2 Single step event at PCGWIN32.0049A4F2
0049E6BE Single step event at PCGWIN32.0049E6BE
00474349 Access violation when writing to [657EAB6E] <<-- 2' blocco di eccezioni
0047456A Access violation when writing to [B4E6507E]
004746DE Single step event at PCGWIN32.004746DE
00474A23 Single step event at PCGWIN32.00474A23
77F65A58 INT3 command at ntdll.DbgBreakPoint
77F65A58 INT3 command at ntdll.DbgBreakPoint
77F65A58 INT3 command at ntdll.DbgBreakPoint
004844F2 Single step event at PCGWIN32.004844F2 <<-- qui mettiamo il Memory Breakpoint on Access
77310000 Module C:\WINDOWS\system32\COMCTL32.dll
76C50000 Module C:\WINDOWS\system32\IMAGEHLP.dll
77BE0000 Module C:\WINDOWS\system32\msvcrt.dll
76360000 Module C:\WINDOWS\system32\comdlg32.dll
70A70000 Module C:\WINDOWS\system32\SHLWAPI.dll
7CCC0000 Module C:\WINDOWS\system32\SHELL32.dll
78090000 Module C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.1612_x-ww_7c379b08\comctl32.dll
004886BE Single step event at PCGWIN32.004886BE
77BD0000 Module C:\WINDOWS\system32\VERSION.dll
5B180000 Module C:\WINDOWS\System32\uxtheme.dll
00488F66 Single step event at PCGWIN32.00488F66 <<-- qui termino il programma
0049EF66 Single step event at PCGWIN32.0049EF66
746B0000 Module C:\WINDOWS\System32\MSCTF.dll
Process terminated, exit code 0
```

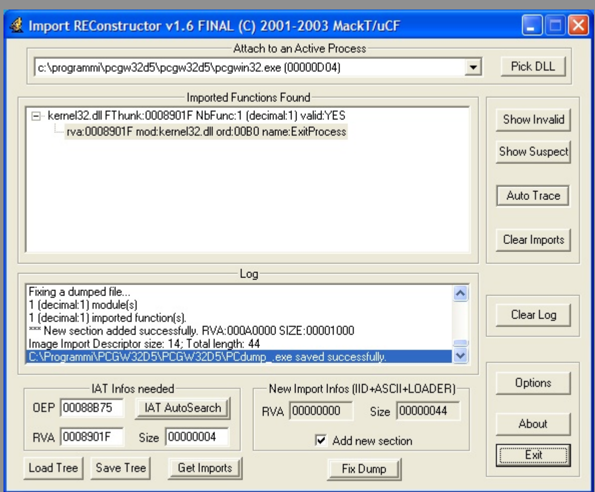
Cosa deduciamo? Dai log di esempio che avevo riportato sopra è ovvio che il programma PCGWIN32.EXE è stato packato 2 volte con PCGuard 5.0!
Il Memory Breakpoint on Access lo dobbiamo mettere dopo la penultima eccezione SingleStep del secondo blocco di eccezioni, come si può vedere nel log. Ed ecco come si presenta il log quando ho trovato l'OEP:

```
IsDebugPresent hidden
0048A349 Access violation when writing to [657EAB6E]
0048A56A Access violation when writing to [B4E6507E]
0048A6DE Single step event at PCGWIN32.0048A6DE
0048AA23 Single step event at PCGWIN32.0048AA23
77F65A58 INT3 command at ntdll.DbgBreakPoint
77F65A58 INT3 command at ntdll.DbgBreakPoint
77F65A58 INT3 command at ntdll.DbgBreakPoint
0049A4F2 Single step event at PCGWIN32.0049A4F2
0049E6BE Single step event at PCGWIN32.0049E6BE
00474349 Access violation when writing to [657EAB6E]
0047456A Access violation when writing to [B4E6507E]
004746DE Single step event at PCGWIN32.004746DE
00474A23 Single step event at PCGWIN32.00474A23
77F65A58 INT3 command at ntdll.DbgBreakPoint
77F65A58 INT3 command at ntdll.DbgBreakPoint
77F65A58 INT3 command at ntdll.DbgBreakPoint
004844F2 Single step event at PCGWIN32.004844F2
77310000 Module C:\WINDOWS\system32\COMCTL32.dll
76C50000 Module C:\WINDOWS\system32\IMAGEHLP.dll
77BE0000 Module C:\WINDOWS\system32\msvcrt.dll
76360000 Module C:\WINDOWS\system32\comdlg32.dll
70A70000 Module C:\WINDOWS\system32\SHLWAPI.dll
7CCC0000 Module C:\WINDOWS\system32\SHELL32.dll
78090000 Module C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.1612_x-ww_7c379b08\comctl32.dll
00488B75 Memory breakpoint when reading [00401000] <<-- ecco l'OEP
```

Ora facciamo il dump del nostro bel processo, in cui possiamo vedere che l'OEP è 88B75:



Poi con ImportRec controlliamo la IAT del dump del processo che ha solo un puntatore non risolto, che culo!



Fixiamo la IAT con la solita ExitProcess e abbiamo il nostro bel PC Guard unpackato!
Mi raccomando non unpackate troppo altrimenti diventate ciechi!

ï¿½½
ï¿½

...::: EPOKH :::...

Note finali

Grazie ad Eloo (un cracker francese) per avermi concesso gentilmente i log particolari di OllyDbg. Per la cronaca: non gli ho chiesto il permesso visto che sono pubblici, però visto che si è rotto le palle a studiarsi PC Guard merita un ringraziamento lo stesso.
Mi sono interessato su PC Guard perchè c'è un programma di nome EGirl (un pornazzo virtuale lo so) che mi ha dato del filo da torcere per un pò.

Disclaimer

Vorrei ricordare che il software va comprato ei¿½ non rubato, dovete registrare il vostro prodotto dopo il periodo di valutazione. Non mi ritengo responsabile per eventuali danni causati al vostro computer determinati dall'uso improprio di questo tutorial. Questo documento ï¿½ stato scritto per invogliare il consumatore a registrare legalmente i propri programmi, e non a fargli fare uso dei tantissimi file crack presenti in rete, infatti tale documento aiuta a comprendere lo sforzo che ogni sviluppatore ha dovuto portare avanti per fornire ai rispettivi consumatori i migliori prodotti possibili.

Reversiamo al solo scopo informativo e per migliorare la nostra conoscenza del linguaggio Assembly.ï¿½